Testbed-12 PubSub / Catalog Engineering Report

Table of Contents

Publication Date: 2017-05-12

Approval Date: 2017-05-11

Posted Date: 2016-12-19

Reference number of this document: OGC 16-137r2

Reference URL for this document: http://www.opengis.net/doc/PER/t12-A074

Category: Public Engineering Report

Editor: Lorenzo Bigagli

Title: Testbed-12 PubSub / Catalog Engineering Report

Testbed-12 PubSub / Catalog Engineering Report (16-137r2)

COPYRIGHT

WARNING

LICENSE AGREEMENT

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER’S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by

destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Abstract

This document describes how the OGC PubSub standard can be used as a mechanism to automatically notify analysts of data availability for CSW and other OGC Web Services (e.g. WFS, WCS). In particular, this document proposes the following:

- Specific PubSub 1.0 extensions for CSW 2.0.2 and 3.0, leveraging on standard functionalities, data models, and semantics to enable sending notifications based on user-specified area of interest and/or keywords;
- A general, basic mechanism for enabling PubSub for the generic OGC Web Service over the existing request/reply OWS's, i.e. usual requests as filters, usual responses as appropriate updates/data pushes, existing semantics and syntax expressiveness.

This document is the result of activity performed within the Large-Scale Analytics (LSA) Thread of the OGC Testbed 12 Interoperability initiative, being identified as document deliverable "A074 PubSub / Catalog Engineering Report". This document also captures lessons learnt from the implementation of component deliverable "A016 CSW 2.0.2 with PubSub Core Support Server".

Business Value

The implementation of efficient, push-based data discovery is necessary to support the transition to advanced push-based service interaction styles and enable the ubiquitous sensor-based applications envisioned by the Internet of Things.

Technology Value

Much work has been conducted in the past (e.g. in previous Testbed Initiatives) on event-based models and architectures. Recently, the PubSub 1.0 standard has introduced an abstract model for Publish/Subscribe message exchange, along with a SOAP binding. However, the current OGC Baseline only supports synchronous web service request-response capabilities. This ER will address this gap and exemplify the use of PubSub, particularly in conjunction with the Catalog Service interface.

How this ER relates to the work of the Working Group

The extensions introduced in this ER are among the first applications of the OGC PubSub 1.0 Specification. Hence, this ER is of great importance to assess and validate the specification itself, in particular as regards the following aspects:

- Extensibility of the PubSub 1.0 specification;
- Experimentation of PubSub 1.0 with a variety of bindings and delivery methods, in particular Server-Sent Events (SSE);
- Use of the OGC Filter Encoding 2.0 language to express subscription filters;
- Pluggability of filter languages and delivery methods.

This ER also contributes to some of the activities in the current scope for the PubSub SWG, such as:

- Definition of a PubSub RESTful binding;
- Definition of a generic mechanism to PubSub-enable the existing OWSs;
- Definition of lexical constants (e.g. “ALL” to identify all the publications offered by a Publisher).

SSE is a widely supported technology and should be considered for standardization in the PubSub SWG. The ER authors are encouraged to formalize a "SSE Delivery Method" Conformance Class and submit it to the SWG for consideration and standardization.

Finally, the ER provides useful indications on future lines of work for the PubSub specification, such as:

- Specific mechanisms for standardizing delivery methods;
- Integration of legacy components in an eventing architecture (cf. the PubSub Brokering Publisher).

Keywords

ogcdocs, testbed-12, Publish/Subscribe, CSW

Proposed OGC Working Group for Review and Approval

The PubSub SWG has agreed to be the primary reviewer and commenter for this Engineering Report. The Catalog DWG and the Catalog Services 3.0 SWG have agreed to act as secondary reviewers.

Chapter 1. Introduction

1.1. Scope

This document addresses PubSub-enabled Catalogs, as a fundamental building block to advance the current OGC standard baseline.

It aims at defining a specific PubSub extension for CSW 2.0.2 and 3.0, as well as a general, simple mechanism for enabling PubSub in the generic OGC Web Service.

This OGC® document is applicable to all the OGC Web Services specifications, in particular to the CSW 2.0.2 and 3.0 Implementation Specifications.

1.2. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Table 1. Contacts

Name	Organization
Lorenzo Bigagli (editor)	CNR
Panagiotis (Peter) A. Vretanos (contributor)	CubeWerx Inc.
Mark Lawrence (contributor)	Compusult
Fabrizio Papeschi (contributor)	CNR
Richard Martell (contributor)	Galdos

1.3. Future Work

Parts of this document may be further elaborated in the future by the relevant SWGs. For example, the PubSub SWG may incorporate (part of) chapter report_PSRB in the current Publish/Subscribe RESTful Binding (PSRB) draft.

1.4. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 2. References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- OGC 06-121r3, OGC® Web Services Common Specification, version 1.1.0 (9 February 2007)
- OGC 13-131, OGC® Publish/Subscribe Interface Standard 1.0 - Core
- OGC 13-133, OGC® Publish/Subscribe Interface Standard 1.0 - SOAP Protocol Binding Extension
- OGC 13-132, OGC® Publish/Subscribe Interface Standard 1.0 - RESTful Protocol Binding Extension (draft, in progress)
- OGC 07-006r1, OpenGIS® Catalogue Services Specification, version 2.0.2, corrigendum 2
- OGC 07-045, OpenGIS® Catalogue Services Specification 2.0.2 - ISO Metadata Application Profile, version 1.0
- OGC 07-110r4, CSW-ebRIM Registry Service - Part 1: ebRIM profile of CSW, version 1.0.1, corrigendum 1
- OGC 07-144r4, CSW-ebRIM Registry Service - Part 2: Basic extension package, version 1.0.1, corrigendum 1
- OGC 08-103r2, CSW-ebRIM Registry Service - Part 3: Abstract Test Suite, version 1.0.1

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r3] shall apply. In addition, the following terms and definitions apply.

3.1. Broker | Brokering Publisher

Intermediary between Subscribers and other Publishers which have been previously registered with the broker. The broker is not the original producer of messages, but only acts as a message middleman, re-publishing messages received from other Publishers and decoupling them from their Subscribers

3.2. Message

A container within which data (such as XML, binary data, or other content) is transported. Messages may include additional information beyond data, including headers or other information used for routing or security purposes

3.3. Publication

A uniquely identified aggregation of messages published by a Publisher over time. A Publisher may offer any number of publications that Subscribers may subscribe to

3.4. Publisher

An entity that offers publications to Subscribers; supports subscription management (subscribe, unsubscribe) and is responsible for filtering and matching messages of interest to active subscriptions

3.5. Receiver

An entity that receives messages from Senders; may (but need not) be the original Subscriber

3.6. Reliable Publisher

A publisher of messages that offers capabilities to detect and recover from message delivery losses, whether caused by network failures, software failures, hardware failures, or other causes

3.7. Sender

Entity that sends messages to Receivers; may (but need not) be the initial creator/producer of the data in the message payload

3.8. Subscriber

Entity that creates a subscription at a Publisher; may (but need not) be the Receiver of delivered messages

3.9. Subscription

Expression of interest in all or part of a publication offered by a Publisher. When a subscription has been created, the Publisher delivers messages that match the subscription criteria to the Receiver defined in the subscription

Chapter 4. Conventions

4.1. Abbreviated terms

- CFP Call for Participation
- DWG Domain Working Group
- HTTP Hypertext Transfer Protocol
- MEP Message Exchange Pattern
- OGC Open Geospatial Consortium
- RFQ Request for Quotation
- SSE Server-Sent Events
- SWG Standards Working Group
- UML Unified Modeling Language
- XML eXtensible Markup Language

4.2. UML notation

Most diagrams that appear in this document are presented using the UML 2 static structure diagram, as described in Subclause 5.2 of [OGC 06-121r9].

All classes in this document are extensible and may be extended with application- or domain-specific content via Extension blocks.

NOTE	The UML shown in this document is considered conceptual and abstract, and should not be interpreted as an implementation strategy for bindings that extend and implement a standard. For example, TM_Instant from ISO 19108 may be used to represent time instants for conceptual clarity, but bindings and implementations of this document may realize TM_Instant as a GML TimeInstant, an ISO 8601 date string, or any other representation that is consistent with TM_Instant.

Chapter 5. Overview

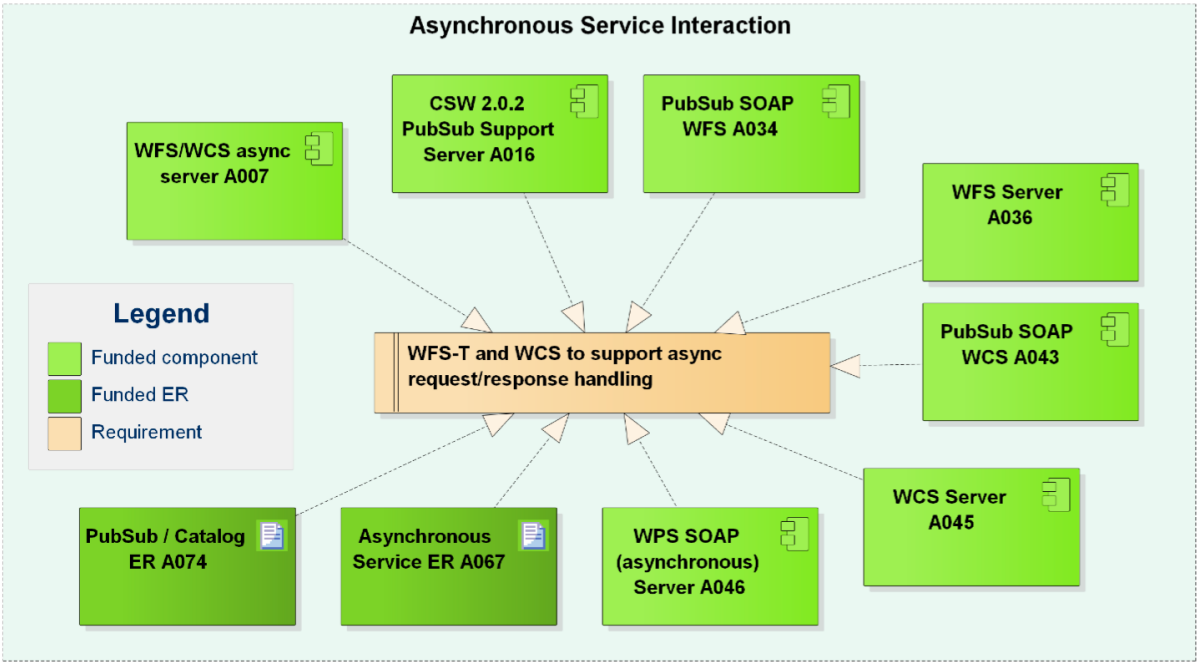
The OGC has conducted significant work on event-based models and architectures in the past, resulting in the Web Notification Service (WNS) Best Practice [1: http://portal.opengeospatial.org/files/?artifact_id=18776], the proposed Sensor Alert and Sensor Event Service, as well as several ERs produced in previous OGC interoperability initiatives. Starting from 2010, these efforts have been subsumed under the scope of the PubSub SWG and resulted in the adoption of the Publish/Subscribe Interface Standard 1.0 in February 2016.

Although the current OGC Baseline still supports only synchronous web service request-response capabilities, there is wide consensus that standard solutions for push-based message exchange patterns are fundamental enablers of the OGC vision, for example to implement the ubiquitous sensor-based applications in the advanced proactive control scenarios envisioned by the Internet of Things.

Several work items in the OGC Testbed 12 RFQ/CFP [2: http://www.opengeospatial.org/standards/requests/139] address the means to incorporate forms of asynchronous service interaction, including Publish/Subscribe message patterns, for example in WPS, WCS, WFS, or in geospatial queries of aviation data.

In particular, the RFQ/CFP includes a specific Asynchronous Service Interaction subtask (see figure Asynchronous Service Interaction subtask), part of a set of subtasks that aim at enhancing the OGC Baseline, by extending OGC architectural designs through efforts that cross over several individual standards and services and are applied in a much wider scope.

Asynchronous Service Interaction subtask



The subtask description in the RFQ/CFP distinguishes among three different approaches to handle asynchronous interaction with OGC Web services:

1. WPS façades;
2. Specific extensions to each OGC Web Service with asynchronous request/response capabilities;

3. OGC PubSub.

The document deliverable "A067 Implementing Asynchronous Service Response Engineering Report" (OGC 16-023) elaborates on the above approaches in situations where big chunks of data require asynchronous delivery. The ER focuses on the first and the second approach, with the goal to summarize and compare the results from using a WPS facade and an extension for WFS for asynchronous service responses, as well as to provide recommendations for future activities.

This document deliverable "A074 PubSub/Catalog Engineering Report" (OGC 16-137) focuses on the third approach, OGC PubSub, in the specific case of catalogs, investigating the functional requirements of an interoperable, push-based data discovery solution.

As underlined in the RFQ/CFP, it is important to provide methods that support notification (push) of new data as opposed to search (pull), given the volume of data that will be available in catalogs in the near future.

The recently approved OGC PubSub standard is intended as an overarching model to extend services with Publish/Subscribe capabilities. The Publish/Subscribe model is distinguished from the request/reply and client/server models by the asynchronous delivery of messages and the ability for a Subscriber to specify an ongoing (persistent) expression of interest.

The PubSub standard is agnostic as regards delivery methods. It defines a Publisher role that may support multiple delivery methods, such as ATOM, AMQP, or SOAP, as advertised in its Capabilities document, in the DeliveryCapabilities section. By implementing the Publisher interface, a PubSub-OWS may offer more than one method of delivery for each Publication, to be chosen by Subscribers. Publish/Subscribe would imply push-style message delivery, however some methods may actually be pull-based (e.g. polling), under the hood.

Hence, both the solutions investigated in ER OGC 16-023, the WPS façade and the specific WFS extension for asynchronous responses, are compatible with the PubSub standard and may be integrated in an application.

The application of OGC PubSub to OGC Catalogue Services allows an analyst to register and be notified when new metadata records become available. Such new OGC-compliant PubSub-enabled discovery service supports and promotes the system architecture of the Testbed 12 Initiative, as a fundamental building block to advance the current OGC standard baseline. As a consequence, this work may contribute to the enhanced availability of standards-based offerings in the marketplace.

In terms of the RFQ/CFP threads, the Asynchronous Service Interaction subtask is assigned to the Large-Scale Analytics (LSA) thread, which addresses short- and long-term planning and analysis of geospatial topics, domains, and questions. The LSA thread includes elements that help the investigator to discover data in an optimal way. Hence, this document is mainly related to the LSA thread. However, it may as well be of interest for the Aviation (AVI) thread (cf. E001 Catalog ER (16-024r2); E003 Asynchronous Messaging ER (16-017r1); F003 CSW component implementation) and the Linked Data and Advanced Semantics (LDS) thread (cf. the Catalog and SPARQL ER (16-062) and related component implementation work items).

The rest of this document is organized as follows:

1. Specific PubSub 1.0 extension for CSW - how to implement the PubSub 1.0 Core Conformance

Classes on top of CSW instances, leveraging on their standard functionalities, data models, and semantics;

2. Basic PubSub 1.0 extension for the generic OWS - a simple way to enable existing request/reply OWS's to Publish/Subscribe, by implementing the Capabilities components required by PubSub 1.0 Core;
3. Report on the PubSub RESTful Binding - how PubSub 1.0 Core operations, encodings and messages are bound to a RESTful service interface;
4. Appendix - Additional XML Schema Documents introduced in this Engineering Report;
5. Appendix - Relevant findings, recommendations, Use Case(s), architectural schemes, Change Request(s);
6. Appendix - Report on prototype PubSub-CSW implementation in LSA-A016: CSW 2.0.2 with PubSub Core Support.

Chapter 6. Specific PubSub 1.0 extension for CSW

This chapter introduces a PubSub extension for CSW. It describes how to implement the PubSub 1.0 Core Conformance Classes (see figure Figure 1) on top of CSW instances, leveraging on their standard functionalities, data models, and semantics.

In particular, a CSW implementing this extension is capable of sending notifications based on user-specified area of interest and/or keywords, as required by the Testbed 12 RFQ.

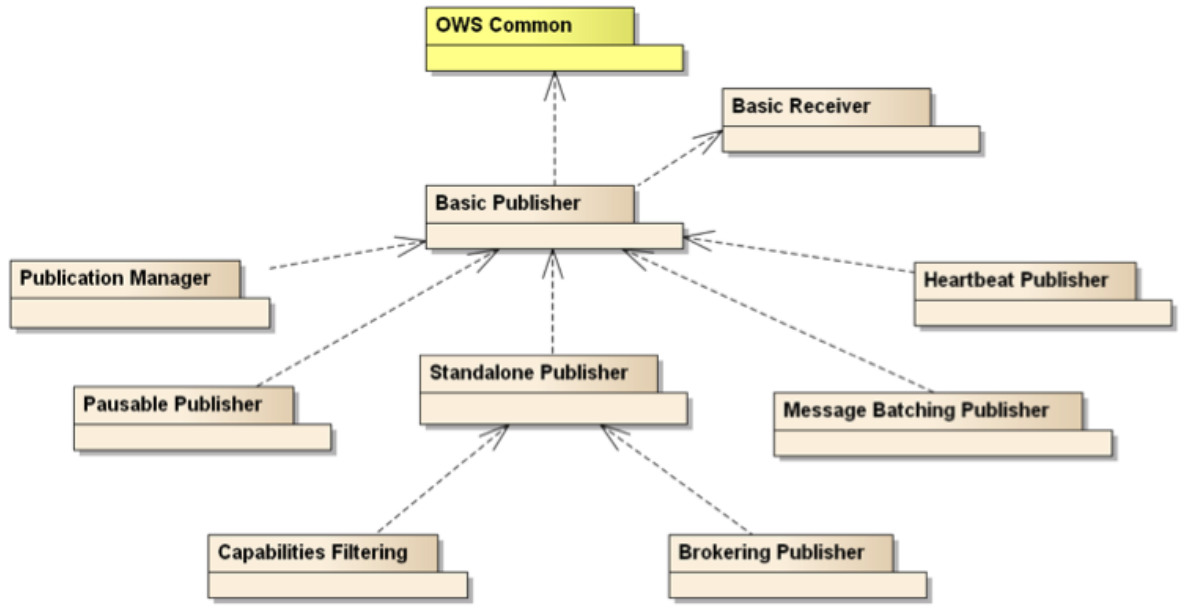


Figure 1. PubSub Core Conformance Classes

The primary target for this extension is CSW 2.0.2, a mature and widespread specification, for which several profiles, bindings and extensions are defined. Possible specific adaptations to CSW 3.0, which has been formally approved in February 2016 and published in June 2016, are considered.

To address interoperability issues in multi-catalog type scenarios, we also consider possible specific adaptations to the ISO Application Profile, the ebRIM Application Profile, the OpenSearch extension, the SOAP binding, and DCAT.

Lastly, this chapter introduces a message delivery method based on Server-Sent Events (SSE), implementing an actual push-based (i.e. not simulated, polling-based) mechanism.

6.1. Conceptual model

The main functionality of a CSW is to hold a set of metadata records on geospatial resources, which can be queried by clients specifying an arbitrary combination of query criteria, e.g. area of interest, keyword, time interval.

Conceivably, a client may be interested in being notified when the record set matching her/his criteria of interest changes, instead of having to repeat the query and look for the differences.

Possible changes in the record set include: new records created, existing records modified (e.g. new/modified/deleted attribute), or deleted.

The PubSub specification is agnostic as to what constitutes a change, i.e. an event that should cause a notification by a Publisher (aka its event model). It is only required that a Publisher instance communicate what notifications it will emit by advertising them in the Publication section of its Capabilities document (see below).

The main use-case supported by CSW's functionalities is geospatial data discovery, for which the main event of interest is the availability of new metadata records in the catalogue. Hence, the extension introduced in this chapter mainly supports the notification of new records matching the client's criteria of interest, specified in his/her subscription. It partially supports the notification of deletions and modifications of existing records, as detailed in chapter Notification encoding. The precise definition of an event model for CSW is left to the relevant OGC Working Groups.

This event model can be implemented by standard mechanisms of the CSW specification, namely the GetRecords and the Transaction operation. The proposed basic Message Exchange Pattern (MEP) for a PubSub-CSW is represented in figure Figure 2 and can be summarized as follows:

1. The CSW client subscribes specifying a GetRecords request to be used as filter for the notifications;
2. The CSW client obtains the Time-0 recordset via a standard Request/Reply GetRecords (same request as above);
3. The PubSub-CSW notifies the client of subsequent updates using the standard CSW TransactionResponse semantics, content model and syntax.

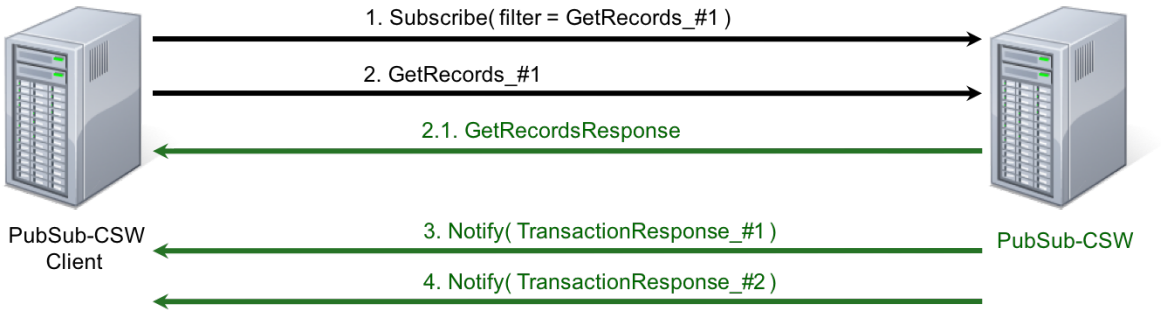


Figure 2. CSW Publish/Subscribe MEP

This MEP allows binding the PubSub 1.0 Core operations, encodings and messages to the standard CSW functionalities, data models, and semantics. In particular, the GetRecords request supports filtering notifications on user-specified area of interest and/or keywords, as per the RFQ/CFP requirements.

6.2. Required Capabilities components

PubSub Core requires that a CSW advertise the implemented Conformance Classes in its Capabilities document, namely in the Profile property of the ServiceIdentification section (as of OWS Common 1.1). Besides, it requires that the additional Capabilities components represented in Figure 2 are returned in the GetCapabilities response, but does not specify the specific mechanism for incorporating these additional Capabilities components into the CSW Capabilities document.

These extension proposes to include these additional Capabilities components in the ExtendedCapabilities of the PubSub-CSW, as detailed in the following chapters.

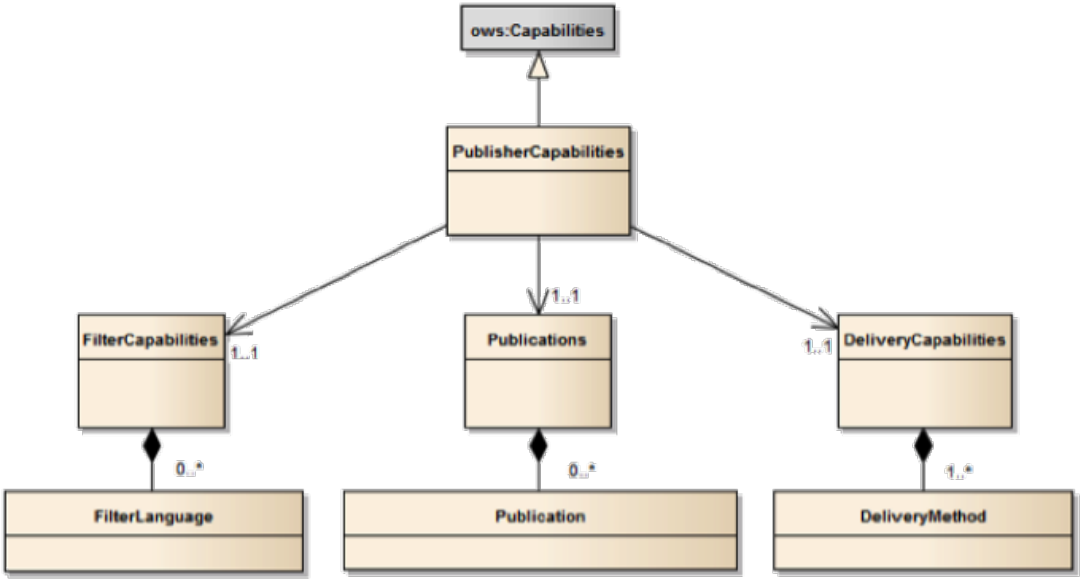


Figure 3. PubSub Capabilities components

6.2.1. FilterCapabilities

The FilterCapabilities section describes the filtering-related capabilities of the PubSub-CSW, i.e. the filter languages it supports for matching events against subscriptions.

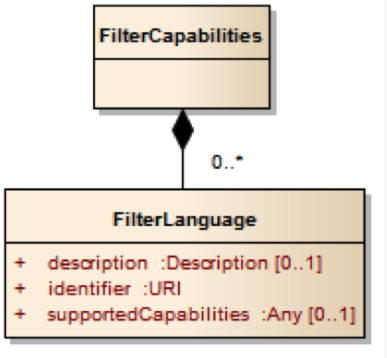


Figure 4. Filter Capabilities

The following Capabilities snippet declares that this PubSub-CSW instance accepts GetRecords requests as subscription filters.

FilterCapabilities

```
<FilterCapabilities>
   <FilterLanguage>
      <Abstract>This PubSub-CSW accepts GetRecords requests as subscription filters.
      </Abstract>
      <Identifier>http://www.opengis.net/cat/csw/3.0/GetRecords
      </Identifier>
   </FilterLanguage>
</FilterCapabilities>
```

6.2.2. DeliveryCapabilities

The DeliveryCapabilities section describes the delivery methods supported by the PubSub-CSW, e.g. SOAP, WS-Notification.

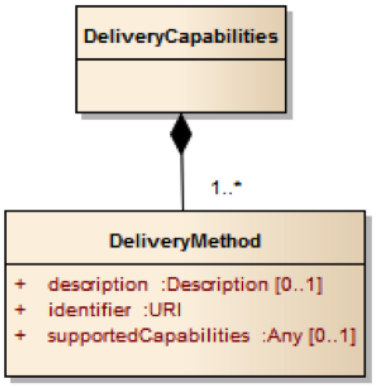


Figure 5. Delivery Capabilities

The following Capabilities snippet declares that this PubSub-CSW instance delivers notifications via SOAP/HTTP, as defined by the PubSub SOAP Binding standard.

DeliveryCapabilities

```
<DeliveryCapabilities>
   <DeliveryMethod>
      <Abstract>This PubSub-CSW supports notification delivery via SOAP/HTTP.
      </Abstract>
      <Identifier>http://www.opengis.net/spec/pubsub/1.0/req/soap/http-delivery-
publisher</Identifier>
   </DeliveryMethod>
</DeliveryCapabilities>
```

6.2.3. Publications

The Publications section describes the contents offered by the PubSub-CSW, i.e. the sequences of notifications that Subscribers can subscribe to.

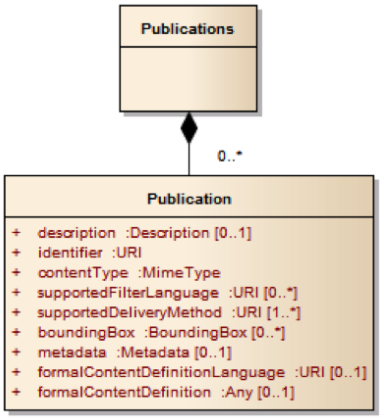


Figure 6. Publications

The following Capabilities snippet declares a publication that notifies on new records from the USGS Earthquake Catalog. Notifications can be filtered with the semantics of the GetRecords request and are delivered as TransactionResponses via SOAP/HTTP.

Publications

```
<Publications>
   <Publication>
      <Abstract>This publication notifies on new records from the USGS Earthquake
Catalog, an implementation of the FDSN Event Web Service Specification, allowing
custom searches for earthquake information using a variety of parameters.</Abstract>
      <Identifier>EARTHQUAKE</Identifier>

<ContentType>http://www.opengis.net/cat/csw/3.0/TransactionResponse</ContentType>
      <SupportedFilterLanguage>http://www.opengis.net/cat/csw/3.0/GetRecords
      </SupportedFilterLanguage>
      <SupportedDeliveryMethod>http://www.opengis.net/spec/pubsub/1.0/req/soap/http-
delivery-publisher</SupportedDeliveryMethod>
   </Publication>
</Publications>
```

Notification encoding

The Catalog Service specification provides the optional Transaction operation, which allows clients to request a specified set of “insert”, “update”, and “delete” actions on the content managed by a Catalogue Service instance.

This operation provides the semantics and the syntax to support the core event model of a CSW, i.e. the availability of new metadata records in the catalogue, as well as their modification and deletion. In particular, the TransactionResponse contains the following elements:

- transactionSummary - a summary that includes the total number of records inserted, updated, and deleted by a transaction;
- insertResults - a brief representation of the records created by a transaction, which includes the record identifier.

By receiving a TransactionResponse, a Subscriber is able to easily retrieve the new records matching the Subscription criteria, and to obtain a summary indication about deleted and modified records. This fully supports the main use-case in geospatial data discovery, and partially supports less common use-cases.

It is worth noting that the syntax of a TransactionRequest could allow a more explicit indication of all inserted/updated/deleted records. However, its specification seems less clear and consolidated. Besides, the normal flow of a TransactionRequest is from a client to a CSW, hence a TransactionResponse seems more straightforward. Future evolutions of this extension may evaluate the use of TransactionRequests for encoding PubSub notifications.

6.3. Service interface

6.3.1. Introduction

Interaction with an OGC web service commences with the retrieval of the service's capabilities document. It is the jumping off point from which a client can access an OGC web service's offerings.

The traditional OGC capabilities document, particularly for data services such as a CSW, is composed of the following sections:

- a Service Identification section
- a Service Provider section
- an Operations Metadata section
- a Content section
- a Filter capabilities section

NOTE	All sections are optional so that they may be retrieved individually, severally or as a complete capabilities document.

The current design of the capabilities document is based on an RPC view of a web service and so it is not obvious how a resource-base service interface might be described by an OGC capabilities document.

Before we consider the question of how to describe a RESTful service interface in an OGC capabilities document, and specifically in the capabilities document of a CSW 3.0 service, we need to consider what the goals of such a description should be. At the very least, a client inspecting an OGC capabilities document should be able to:

a) Determine if the service offers a RESTful service interface b) Determine which resources the services offers and their URLs

6.3.2. Availability of a RESTful service interface

There are several approaches that might be taken in order to allow a service to advertise the availability of a RESTful service interface.

Approach 1: Conformance classes

OGC web service specifications define one or more conformance classes that encapsulate basic units of capability that someone implementing the service might offer. Recent practice in OGC has been to include, in the capabilities document, some means of allowing a service to explicitly declare which conformance classes a service implements.

Thus, an obvious approach for advertising the availability of a RESTful service interface is for a service specification to define one or more conformance classes encapsulating the elements of the interface. A client inspecting a compliant server's capabilities document can then explicitly determine if the service offers a RESTful interface or not by inspecting the conformance class declarations.

Approach 2: Hypermedia controls

Another approach for signaling to a client that a service offers a RESTful interface is the existence of hypermedia controls in the capabilities document. This is the approach taken by the WMTS with its ResourceURL element and the WFS 2.5 with its use of hypermedia controls encoded as ATOM links. The presence of ATOM links pointing to offered resources implicitly signals the availability of a RESTful service interface.

Approach 3: Alternative service description document

A third approach would be to define a specific element in the capabilities document that points to an alternate, perhaps specialized, description document for the service. This is the approach that was used in the CSW capabilities document, via an ows:Constraint named "WSDL", to point to a WSDL description document. A similar approach could be taken for a RESTful interface; a specific element or constraint might be defined that points to a REST-specific interface description document such a Swagger or RSDL document. The existence of this element in capabilities document would signal to a client that a REST interface is available.

It should be noted that these approaches are not mutually exclusive and one or more could be used simultaneously in a capabilities document to signal the availability of a RESTful interface.

6.3.3. Resources and resource URL

There are several ways that a service, in its capabilities document, might advertise which resources it offers and the URLs for those resource.

Predefined resource URLs

The first approach is for the service specification to describe predefined resource paths that a service may offer. These may be tied to specific conformance classes and declaring that a service implements such a conformance class implies that the service offers that resource. A current draft specification, OGC® Publish/Subscribe Interface Standard 1.0 RESTful Protocol Binding Extension, defines the "RESTful Basic Publisher" conformance class. A service declaring that it implements this conformance class is implicitly declaring that it offers the */publications* and */subscriptions* resources.

Hypermedia controls and opaque URLs

Hypermedia controls are simply links embedded in a service's response that allow a client to determine which next states are available. This is analogous to a web pages containing hyperlinks that let a viewer know to which other pages or content they may move. Hypermedia controls are encoded to include the URL of the target resource and also include an association type that describes the relationship between the current resource and the target resource. Thus, hypermedia controls in an OGC web service's capabilities document may be used to let a client know which other resources are offered by the service and what the relationship is between the service and those resources. The following XML fragment from a WFS service that implements the draft WFS 2.5 specification is an example of how hypermedia controls are used in the description of a feature type to advertise the resource's access URL and the resource's schema:

WFS FeatureType Example

```
    <FeatureType>
      <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                 rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wfs/2.5.0/USGS/SfBuildings"/>
      <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                 rel="describedBy"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wfs/2.5.0/USGS/schema/SfBuilding
s"/>
      <Name>cw:SfBuildings</Name>
      <Title>San Francisco Buildings</Title>
      <DefaultCRS>urn:ogc:def:crs:EPSG::4326</DefaultCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::3857</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::3395</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::4267</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::4269</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::26709</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::26710</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::26711</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::26712</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::26909</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::26910</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::26911</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::26912</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::32609</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::32610</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::32611</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::32612</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:OGC::CRS41001</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::42101</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::42103</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::42105</OtherCRS>
      <OtherCRS>urn:ogc:def:crs:EPSG::102002</OtherCRS>
      <ows:WGS84BoundingBox>
        <ows:LowerCorner>-122.514259401308 37.70524337916189</ows:LowerCorner>
        <ows:UpperCorner>-122.357630553475 37.8319653651556</ows:UpperCorner>
      </ows:WGS84BoundingBox>
    </FeatureType>
```

Notice that two hypermedia controls, encoded as atom:link elements, are included in this example. The first one, with *rel="collection"* provides the base URL of the collection of San Francisco building footprint features. The second one, with *rel="describedBy"* provides the URL to the schema of the building footprint feature type. In all cases the resource URLs are opaque to the client.

NOTE	The schema for the atom:link element is defined in the OGC KML specification (see OGC 12-07r2). Unlike XLinks, atom:link elements allow the URL of a resource to be specified as well as the relationship of that resource to the offering service.

NOTE	Hypermedia controls may also be encoded in the HTTP headers using the *Link* header. Using the headers has the advantage of not requiring modifications to the existing capabilities schema. However, as the number of resources grows, the size of the HTTP header becomes more awkward to manage. Furthermore, link encoded in the header are removed from their context within the capabilities document which tends to obfuscate their meaning and purpose.

URL templates

In some cases the opaque URL approach for pointing to the resources that a service offers is just not feasible. Such is the case for a WMTS where literally billions of URLs would need to be listed to point to the tiles that the service offers. In this case, rather than using opaque URLs, using URL templates makes more sense. URL templates provide a recipe, using substitution variables, for composing the URL to any resource that the service offers. The approach taken by the WMTS standard is to define the *ResourceURL* element that is used to advertise the URL template for each matrix set that the service offers:

WMTS Layer Example

```
   <Layer>
     <ows:Title xml:lang="en">National Land Cover</ows:Title>
     <ows:WGS84BoundingBox>
       <ows:LowerCorner>-127.9521005274938 30.57146092341947</ows:LowerCorner>
       <ows:UpperCorner>-113.3506260963461 45.01357346325526</ows:UpperCorner>
     </ows:WGS84BoundingBox>
     <ows:Identifier>National_Land_Cover.National_Land_Cover</ows:Identifier>
     <ows:BoundingBox crs="urn:ogc:def:crs:EPSG::42303">
       <ows:LowerCorner>-2493045 1317885</ows:LowerCorner>
       <ows:UpperCorner>-1713555 2497245</ows:UpperCorner>
     </ows:BoundingBox>
     .
     .
     .
     <TileMatrixSetLink>
       <TileMatrixSet>3395</TileMatrixSet>
     </TileMatrixSetLink>
     <TileMatrixSetLink>
       <TileMatrixSet>smerc</TileMatrixSet>
     </TileMatrixSetLink>
     <ResourceURL format="image/x-jpegorpng"
                  resourceType="tile"
                  template="https://tb12-
1.cubewerx.com/a042/OpenImageMap/tilesets/USGS/National_Land_Cover/National_Land_Cover
/default/{TileMatrixSet}/{TileMatrix}/{TileRow}/{TileCol}.jop"/>
     <ResourceURL format="image/x-jpegorpng"
                  resourceType="simpleProfileTile"
                  template="https://tb12-
1.cubewerx.com/a042/OpenImageMap/tilesets/USGS/National_Land_Cover/National_Land_Cover
/default/smerc/{TileMatrix}/{TileRow}/{TileCol}.jop"/>
     .
     .
     .
     <ResourceURL format="application/json"
                  resourceType="TileJSON"

template="https://tb12.cubewerx.com/a042/cubeserv/default/wmts/1.0.0/tileJSON/National
_Land_Cover.National_Land_Cover"/>
   </Layer>
```

It should be noted that these approaches are not mutually exclusive and one or more could be used simultaneously in a capabilities document to advertise the resources that a RESTful service offers.

6.4. CSW capabilities document with REST extensions

This clause proposes modifications to the existing CSW 3.0 capabilities document schema to allow a client to determine if a RESTful service interface is available and to determine which resources a

PubSub CSW service offers.

Unlike an RPC-based system, in a resource-based system there is no need to explicitly describe operations (i.e. the verbs) because the set of operations is fixed by the HTTP protocol (i.e. HEAD, OPTIONS, GET, POST, PUT and DELETE). As such, there is no need for an Operations metadata section to exist in the capabilities document of a service that only implements a RESTful service interface.

Generating a valid capabilities document without an operations metadata section is possible since the cardinality of the ows:OperationMetadata element is minOccurs=0.

The following XML-Schema fragment adds three new elements to the existing CSW 3.0 capabilities schema:

CSW Capabilities Schema

```
<xsd:element name="Capabilities"
             type="csw30:CapabilitiesType" id="Capabilities"/>
<xsd:complexType name="CapabilitiesType" id="CapabilitiesType">
   <xsd:annotation>
      <xsd:documentation>
         This type extends ows:CapabilitiesBaseType defined in OGC 06-121r9
         to include information about supported OGC filter components. A
         profile may extend this type to describe additional capabilities.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:complexContent>
      <xsd:extension base="ows:CapabilitiesBaseType">
         <xsd:sequence>
            <xsd:element name="ServiceConstraints"
                         type="csw30:ConstraintListType"
                         minOccurs="0"/>
            <xsd:element name="Conformance"
                         type="csw30:ConstraintListType"
                         minOccurs="0"/>
            <xsd:element name="Content"
                         type="csw30:ContentType"
                         minOccurs="0"/>
            <xsd:element ref="fes:Filter_Capabilities" minOccurs="0"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ConstraintListType">
   <xsd:sequence>
      <xsd:choice>
         <xsd:element ref="ows:Constraint"
                      minOccurs="0" maxOccurs="unbounded"/>
         <xsd:element ref="ows:Parameter"
                      minOccurs="0" maxOccurs="unbounded"/>
      <xsd:choice>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="ContentType">
   <xsd:sequence>
      <xsd:element ref="atom:link" minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

The first two elements, csw30:ServiceConstraints and csw30:Conformance, are used to decouple the conformance and constraint information which is currently encoded in the ows:OperationsMetadata section using ows:Constraint and ows:Parameter elements. Embedding this information in the ows:OperationsMetadata section presents a problem for RESTful services because the operations metadata section must include at least 2 ows:Operation elements and as has

already been mentioned, the ows:OperationsMetadata section is not required for a RESTful-only service.

The third element, csw30:Content, is simply a list of hypermedia controls that point to the resources that the service offers (e.g. the catalogue record collections). All RESTful CSW implementations shall include at least one hypermedia control that points to the collection of csw:Record records. Profiles of the CSW, such as the CSW:ebRIM profile, may include additional resources that point to the resource collections from the additional information model(s) offered.

The following example illustrates the capabilities document from a pubsub-enabled CSW that only implements a REST binding.

PubSub CSW Capabilities Example

```
   <?xml version="1.0" encoding="UTF-8"?>
   <csw:Capabilities
      version="3.0.0"
      xmlns="http://www.opengis.net/cat/csw/3.0"
      xmlns:csw="http://www.opengis.net/cat/csw/3.0"
      xmlns:fes="http://www.opengis.net/fes/2.0"
      xmlns:ows20="http://www.opengis.net/ows/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
                          http://schemas.opengis.net/csw/3.0/cswAll.xsd
                          http://www.w3.org/1999/xlink
                          http://www.w3.org/1999/xlink.xsd">
      <ows20:ServiceIdentification>
         <ows20:Title>Catalogue Service for Spatial Information</ows20:Title>
         <ows20:Abstract>terraCatalog 3.2 based OGC CSW 3.0 Catalogue Service for OGC
core and ISO metadata (describing geospatial services, datasets and
series)</ows20:Abstract>
         <ows20:Keywords>
            <ows20:Keyword>OGC</ows20:Keyword>
            <ows20:Keyword>CSW</ows20:Keyword>
            <ows20:Keyword>Catalog Service</ows20:Keyword>
            <ows20:Keyword>metadata</ows20:Keyword>
            <ows20:Keyword>CSW</ows20:Keyword>
            <ows20:Type
codeSpace="http://www.someGeospatialVocabulary.com">theme</ows20:Type>
         </ows20:Keywords>
         <ows20:ServiceType>CSW</ows20:ServiceType>
         <ows20:ServiceTypeVersion>3.0.0</ows20:ServiceTypeVersion>
      </ows20:ServiceIdentification>
      <ows20:ServiceProvider>
         <ows20:ProviderName>con terra GmbH</ows20:ProviderName>
         <ows20:ProviderSite xlink:type="simple"
            xlink:href="http://www.conterra.de"/>
         <ows20:ServiceContact>
            <ows20:IndividualName/>
            <ows20:PositionName/>
```

```
            <ows20:ContactInfo>
               <ows20:Phone>
                  <ows20:Voice>+49-251-7474-400</ows20:Voice>
                  <ows20:Facsimile>+49-251-7474-100</ows20:Facsimile>
               </ows20:Phone>
               <ows20:Address>
                  <ows20:DeliveryPoint>Marting-Luther-King-Weg
24</ows20:DeliveryPoint>
                  <ows20:City>Muenster</ows20:City>
                  <ows20:AdministrativeArea>NRW</ows20:AdministrativeArea>
                  <ows20:PostalCode>48165</ows20:PostalCode>
                  <ows20:Country>Germany</ows20:Country>

<ows20:ElectronicMailAddress>conterra@conterra.de</ows20:ElectronicMailAddress>
               </ows20:Address>
               <ows20:OnlineResource xlink:href="mailto:conterra@conterra.de"/>
            </ows20:ContactInfo>
         </ows20:ServiceContact>
      </ows20:ServiceProvider>
      <csw:ServiceConstraints>
         <ows20:Parameter name="ElementSetNames">
            <ows20:AllowedValues>
               <ows20:Value>brief</ows20:Value>
               <ows20:Value>summary</ows20:Value>
               <ows20:Value>full</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Parameter>
         <ows20:Constraint name="CoreQueryables">
            <ows20:AllowedValues>
               <ows20:Value>Title</ows20:Value>
               <ows20:Value>Subject</ows20:Value>
               <ows20:Value>Abstract</ows20:Value>
               <ows20:Value>AnyText</ows20:Value>
               <ows20:Value>Type</ows20:Value>
               <ows20:Value>Identifier</ows20:Value>
               <ows20:Value>Modified</ows20:Value>
               <ows20:Value>TemporalExtent</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Constraint>
         <ows20:Constraint name="CoreSortables">
            <ows20:AllowedValues>
               <ows20:Value>Title</ows20:Value>
               <ows20:Value>Type</ows20:Value>
               <ows20:Value>Modified</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Constraint>
         <ows20:Constraint name="DefaultSortingAlgorithm">
            <ows20:AllowedValues>

<ows20:Value>http://www.sdisuite.de/terraCatalog/documentation/descriprionOfSortalgori
thm.html</ows20:Value>
```

```
            </ows20:AllowedValues>
         </ows20:Constraint>
         <ows20:Constraint name="FedeartedCatalogues">
            <ows20:AllowedValues>
               <ows20:Value/>
            </ows20:AllowedValues>
         </ows20:Constraint>
         <ows20:Constraint name="OpenSearch">
            <ows20:AllowedValues>
               <ows20:Value>http://www.sdisuite.de/terraCatalog</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Constraint>
      </csw:ServiceConstraints>
      <csw:Conformance>
         ...
         <!-- RESTful Catalogue binding? -->
         <ows20:Constraint name="RESTCatalogueBinding">
            <ows20:AllowedValues>
               <ows20:Value>true</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Constraint>
         ...
         <!-- RESTful Basic Publisher? -->
         <ows20:Constraint name="RESTfulBasicPublisher">
            <ows20:AllowedValues>
               <ows20:Value>true</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Constraint>
         ...
         <!-- Support for OpenSearch query parameters? -->
         <ows20:Constraint name="OpenSearch">
            <ows20:AllowedValues>
               <ows20:Value>true</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Constraint>
         ...
         <!-- CSW-Response support? -->
         <ows20:Constraint name="CSW-Response">
            <ows20:AllowedValues>
               <ows20:Value>true</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Constraint>
         <!-- ATOM-response support? -->
         <ows20:Constraint name="ATOM-response">
            <ows20:AllowedValues>
               <ows20:Value>true</ows20:Value>
            </ows20:AllowedValues>
         </ows20:Constraint>
      </csw:Conformance>
      <csw:Content>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
```

```
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/csw/Records"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/Association"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/AuditableEvent"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/AffectedObject"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/Classification"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/ClassificationNode
"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/ClassificationSche
me"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/ExternalIdentifier
"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/ExternalLink"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/ExtrinsicObject"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/Federation"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/Organization"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
```

```
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/RegistryPackage"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/Registry"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/Service"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/ServiceBinding"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/SpecificationLink"
/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
 00                   rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/Subscription"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/AdhocQuery"/>
        <atom:link xmlns:atom="http://www.w3.org/2005/Atom"
                    rel="collection"

href="https://tb12.cubewerx.com/a037/cubeserv/default/wrs/3.0/ebRIM/User"/>
      </csw:Content>
      <fes:Filter_Capabilities xmlns:ows11="http://www.opengis.net/ows/1.1">
      ...
      </fes:Filter_Capabilities>
   </csw:Capabilities>
```

A CSW client can check the constraints named *RESTCatalogueBinding* and *RESTfulBasicPublisher* to see of the service offers a RESTful interface. The csw:Content section contains the list of additional resources that are available (in addition to any resource implicitly defined by the specification).

6.4.1. OPTIONS method

The HTTP OPTIONS method may be used on any resource to determine what HTTP methods may be used with the resource and what representations are available. It is presented here because the OPTIONS methods may be used to interrogate resources to determine information beyond what might be presented in a capabilities document. The following sequence diagram illustrate the use of the OPTIONS method.

Example 1: Determine what options are available for the capabilities document.

```
CLIENT                                                  SERVER
|                                                         |
|   OPTIONS / HTTP/1.1                                    |
|   Host: www.someserver.com <http://www.someserver.com>  |
|-------------------------------------------------------->|
|                                                         |
|   HTTP/1.1 200 OK                                       |
|   Allow: OPTIONS, GET                                   |
|   Accept: application/xml, text/html                    |
|<--------------------------------------------------------|
```

In this case, the available methods are the OPTIONS and GET methods and the available representations are XML or HTML.

Example 2: Determine what options are available for the csw:Record resource.

```
CLIENT                                                  SERVER
|                                                         |
|   OPTIONS /Record HTTP/1.1                              |
|   Host: www.someserver.com <http://www.someserver.com>  |
|-------------------------------------------------------->|
|                                                         |
|   HTTP/1.1 200 OK                                       |
|   Allow: OPTIONS, GET                                   |
|   Accept: application/xml, text/xml, application/atom+xml |
|<--------------------------------------------------------|
```

In this case, the available methods are the OPTIONS and GET methods and the resource may be represented as XML or ATOM. It should be noted that for this server, the */Record* resource is a read-only resource.

Example 3: Determine what options are available to the */ExtrinsicObject* resource.

```
CLIENT                                                  SERVER
|                                                         |
|   OPTIONS /Record HTTP/1.1                              |
|   Host: www.someserver.com <http://www.someserver.com>  |
|-------------------------------------------------------->|
|                                                         |
|   HTTP/1.1 200 OK                                       |
|   Allow: OPTIONS, GET, POST, PUT, DELETE                |
|   Accept: application/xml, text/xml, application/atom+xml |
|<--------------------------------------------------------|
```

In this case, the available methods are OPTIONS, GET, POST, PUT and DELETE which indicates the

new instanced of an ExtrinsicObject may be created, modified and deleted. The available representations are XML and ATOM.

6.5. Specific adaptations

This chapter describes specific adaptations for CSW 2.0.2, 3.0, and related specifications.

6.5.1. CSW 2.0.2

CNR has experimented with the implementation of PubSub on top of a CSW 2.0.2 instance (see also Appendix Prototype).

Basic Publisher requires (Req 2) that *a Publisher advertise the conformance classes which are supported by the server. Each supported conformance class shall be identified by a unique value of the Profile property of the ServiceIdentification section of the capabilities document, and the Publisher shall pass all tests defined for each listed conformance class*.

However, CSW 2.0.2 depends on OWS Common 1.0.0, which does not define the Profile property in ServiceIdentification. To work around this issue, a CSW 2.0.2 shall include an additional OWS Common 1.1 ServiceIdentification block in the ExtendedCapabilities, as exemplified below.

ServiceIdentification

```
<ExtendedCapabilities xmlns:ows1_1="http://www.opengis.net/ows/1.1"
xmlns:pubsub="http://www.opengis.net/pubsub/1.0">
   <ows1_1:ServiceIdentification>
      <ows1_1:ServiceType>CSW</ows1_1:ServiceType>
      <ows1_1:ServiceTypeVersion>2.0.2</ows1_1:ServiceTypeVersion>
      <ows1_1:Profile>http://www.opengis.net/spec/pubsub/1.0/conf/rest/basic-publisher
                  http://www.opengis.net/spec/pubsub/1.0/conf/core/basic-publisher
      </ows1_1:Profile>
   </ows1_1:ServiceIdentification>
...
```

ISO Application Profile

CNR has experimented with the implementation of PubSub on top of a CSW-ISO instance (see also Appendix Prototype). No specific adaptation was found to be needed to implement PubSub in conjunction with the ISO Application Profile content model.

OASIS ebRIM Application Profile

Compusult has experimented with the implementation of PubSub on top of a CSW-ebRIM instance. The rim:Subscription and the rim:AdhocQuery objects have proved useful to implement the PubSub Subscribe operation and the PubSub Subscription concept, as detailed in chapter CSW3.0.

A9 OpenSearch extension

CNR has experimented with the implementation of PubSub in conjunction with the OpenSearch

extension for CSW 2.0.2 (see also Appendix Prototype). OpenSearch templates have proved a useful syntax for the SupportedCapabilities element of a FilterLanguage, to describe restrictions of the filter expressions allowed in Subscriptions, as exemplified in chapter OWSFilterCapabilities.

SOAP binding

Compusult has experimented with the implementation of PubSub on top of a CSW based on the SOAP binding, by implementing the SOAP Basic Publisher Conformance Class (see also chapter CSW3.0). No specific adaptation was found to be needed to implement PubSub in conjunction with the SOAP binding.

W3C DCAT

W3C DCAT is an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web. The use of DCAT to describe datasets increases discoverability and enables applications to consume metadata from multiple catalogs. It also supports the implementation of distributed queries across federated catalogs.

DCAT is complementary to the actual catalog content model, hence no specific adaptation is anticipated to implement PubSub alongside DCAT. Future research may evaluate the use of DCAT for notification encoding.

6.5.2. CSW 3.0

Compusult has experimented with the implementation of PubSub on top of a CSW 3.0 instance, implementing a PubSub adapter conforming to the SOAP Basic Publisher Conformance Class defined in the PubSub SOAP Binding extension.

Basic Publisher requires (Req 2) that *a Publisher advertise conformance classes which are supported by the server. Each supported conformance class shall be identified by a unique value of the Profile property of the ServiceIdentification section of the capabilities document, and the Publisher shall pass all tests defined for each listed conformance class.*

Since CSW 3.0 is based on OWS Common 2.0, this is easily accomplished:

```
<ows20:ServiceIdentification>
    <Profile>http://www.opengis.net/spec/pubsub/1.0/conf/soap/basic-
publisher</Profile>
</ows20:ServiceIdentification>
```

The following snippet is a Subscribe request (mapped to the WS-BaseNotification NotificationProducer Subscribe operation), with an OGC Filter specifying a keyword and an area of interest.

Subscribe request

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsn="http://docs.oasis-open.org/wsn/b-2"
```

```
xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:fes="http://www.opengis.net/fes/2.0"
    xmlns:gml="http://www.opengis.net/gml/3.2"
    xmlns:pubsub="http://www.opengis.net/pubsub/1.0"
    xsi:schemaLocation="http://schemas.xmlsoap.org/soap/envelope/
http://schemas.xmlsoap.org/soap/envelope/
    http://docs.oasis-open.org/wsn/b-2 http://docs.oasis-open.org/wsn/b-2.xsd
    http://www.w3.org/2005/08/addressing http://www.w3.org/2006/03/addressing/ws-
addr.xsd
    http://www.opengis.net/fes/2.0 http://schemas.opengis.net/filter/2.0/filterAll.xsd
    http://www.opengis.net/gml/3.2 http://schemas.opengis.net/gml/3.2.1/gml.xsd
    http://www.opengis.net/pubsub/1.0
http://schemas.opengis.net/pubsub/1.0/pubsubAll.xsd">
    <soap:Header>
        <wsa:Action>http://docs.oasis-open.org/wsn/bw-
2/NotificationProducer/SubscribeRequest</wsa:Action>
    </soap:Header>
    <soap:Body>
        <wsn:Subscribe>
            <!--
                Identifier and location for the message consumer (receiver).  Messages
(notifications) will be delivered here
            -->
            <wsn:ConsumerReference>
                <wsa:Address>mailto:somebody@someplace.com</wsa:Address>
            </wsn:ConsumerReference>
            <wsn:Filter>
                <wsn:MessageContent Dialect="http://www.opengis.net/fes/2.0">
                    <fes:Filter>
                        <fes:And>
                            <fes:PropertyIsLike escapeChar="\" singleChar="_"
wildCard="%">
                                <fes:ValueReference>csw:AnyText</fes:ValueReference>
                                <fes:Literal>weather%</fes:Literal>
                            </fes:PropertyIsLike>
                            <fes:Intersects>

<fes:ValueReference>ows:BoundingBox</fes:ValueReference>
                                <gml:Envelope srsName="urn:fes:def:crs:EPSG::4326">
                                    <gml:lowerCorner>13.0983 31.5899</gml:lowerCorner>
                                    <gml:upperCorner>35.5472 42.8143</gml:upperCorner>
                                </gml:Envelope>
                            </fes:Intersects>
                        </fes:And>
                    </fes:Filter>
                </wsn:MessageContent>
            </wsn:Filter>
            <wsn:InitialTerminationTime>2016-12-
11T00:00:00Z</wsn:InitialTerminationTime>
            <!-- Required for subscription creation -->
```

```
            <pubsub:PublicationIdentifier>urn:uuid:7f9vc445-9c3a-21d9-8679-
0400200c9a54</pubsub:PublicationIdentifier>
            <!-- The requested mime type for the published contents -->
            <pubsub:ContentType>text/xml</pubsub:ContentType>
        </wsn:Subscribe>
    </soap:Body>
</soap:Envelope>
```

When creating the Subscription, the PubSub-CSW maps it to a rim:Subscription object, which in turn is mapped to a rim:AdhocQuery object containing the filter from the PubSub Subscription.

The rim:AdhocQuery is then executed periodically against the data published in the catalog, using the filter criteria and a "last run date", to ensure only the latest new and updated records are retrieved.

To optimize the performance of the PubSub-CSW (the overarching Testbed 12 thread for this activity is Large-Scale Analytics, focussing on how to handle large amounts of data), it is recommended to leverage the CSW 3.0 requestId mechanism, in combination with the SOAP interface, and the Message Transmission Optimization Mechanism (MTOM), in combination with XML-binary optimized packaging.

In practice, when the CSW executes the rim:AdhocQuery and discovers new and/or updated records, their RegistryObject identifiers are cached and labeled with a given auto-generated UUID (or with the id of the rim:AdhocQuery). This UUID will then be communicated to the Subscriber in the requestId attribute of the TransactionResponse.

```
<TransactionResponse>
    <TransactionSummary requestId="SomeUUID">
        <totalInserted>3</totalInserted>
        <totalUpdated>5</totalUpdated>
        <totalDeleted>0</totalDeleted>
    </TransactionSummary
</TransactionResponse>
```

Clients can then use this requestId in subsequent GetRecords requests, to retrieve the matching records:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<csw:GetRecords xmlns:rim="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gml="http://www.opengis.net/gml"
    xmlns:csw="http://www.opengis.net/cat/csw/3.0" maxRecords="500"
    requestId="SomeUUID"
    outputFormat="application/xml" outputSchema="http://www.isotc211.org/2005/gmd"
    service="CSW" startPosition="1" version="3.0.0"
    xsi:schemaLocation="http://www.opengis.net/cat/csw/3.0
http://schemas.opengis.net/csw/3.0.0/cswGetRecords.xsd http://www.opengis.net/gml/3.2
http://schemas.opengis.net/gml/3.2.1/gml.xsd">
    <csw:Query typeNames="rim:RegistryObject">
        <csw:ElementSetName typeNames="rim:RegistryObject">full</csw:ElementSetName>
    </csw:Query>
</csw:GetRecords>
```

Because the requestId returned in the TransactionResponse was used to cache the records during the initial internal query, the CSW does not need to query the database again, but simply returns the objects associated with the cached identifiers.

Chapter 7. Basic PubSub 1.0 extension for the generic OWS

This chapter introduces a PubSub extension for the generic OWS. This extension is conceived as a simple way to enable the existing request/reply OWS specifications to Publish/Subscribe, by implementing the OGC Publish/Subscribe Interface Standard 1.0.

An OWS implementing this extension is capable of accepting its usual requests as filters, and of sending notifications about data/metadata updates, based on its existing semantics and syntax expressiveness.

7.1. Conceptual model

This chapter describes how PubSub 1.0 Core operations, encodings and messages are modeled in terms of the functionalities of the generic OWS. No assumption is made on the capabilities of the target OWS, other than those defined by the OGC Web Services Common Standard. Hence this extension may apply, for example, to WFS, WCS, and other OWS interfaces.

The PubSub specification is agnostic as to what constitutes a change, i.e. an event that should cause a notification by a Publisher (aka its event model). It is only required that a Publisher instance communicate what notifications it will emit by advertising them in the Publication section of its Capabilities document (see below).

In general, a PubSub-OWS may be able to notify about changes to any component of its information set. For example, it may notify about changes to its Capabilities document. The extension introduced in this chapter addresses the most general case, at the expenses of efficiency and semantic accuracy. The precise definition of an event model for the various OWS's is left to the relevant OGC Working Groups.

The basic PubSub-CSW MEP introduced in the previous chapter can be generalized as follows (see figure Figure 4):

1. The OWS client subscribes specifying a request to be used as filter for the notifications;
2. The OWS client obtains the Time-0 response via a standard Request/Reply, with the same request as above;
3. The OWS notifies the client of subsequent updates to the response, according to its existing semantics and syntax.

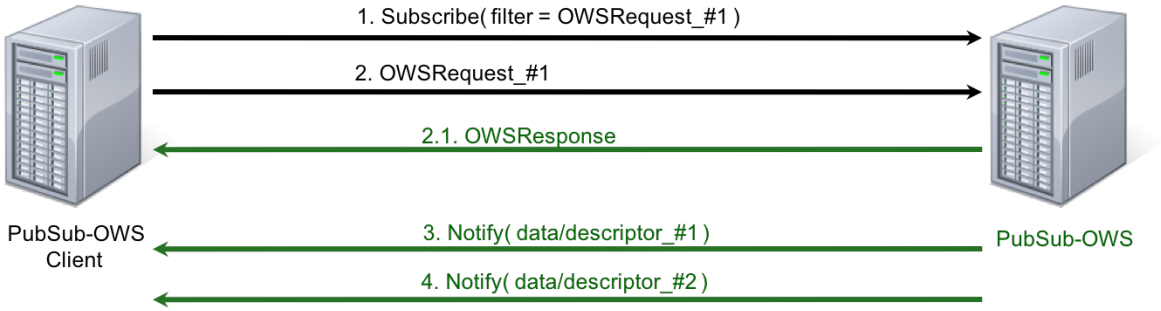


Figure 7. OWS Publish/Subscribe MEP

This may be formalized in an “OWS Request/Reply Publisher” Conformance Class that:

- Accepts OWS requests as subscription filters
  - The Publisher may constraint the filter expressions allowed in Subscriptions (e.g. by imposing OpenSearch templates)
- Sends corresponding OWS responses to notify about data/metadata updates

This MEP is a simple way to enable existing OWSs to PubSub, allowing to bind the PubSub 1.0 Core operations, encodings and messages to the standard OWS functionalities, data models, and semantics.

7.2. Required Capabilities components

PubSub Core requires that the OWS advertise the implemented Conformance Classes in its Capabilities document, namely in the Profile property of the ServiceIdentification section (as of OWS Common 1.1). Besides, it requires that the additional Capabilities components represented in figure Figure 2 are returned in the GetCapabilities response, but does not specify the specific mechanism for incorporating these additional Capabilities components into the OWS Capabilities document. These extension proposes to include these additional Capabilities components in the ExtendedCapabilities of the OWS, as detailed in the following chapters.

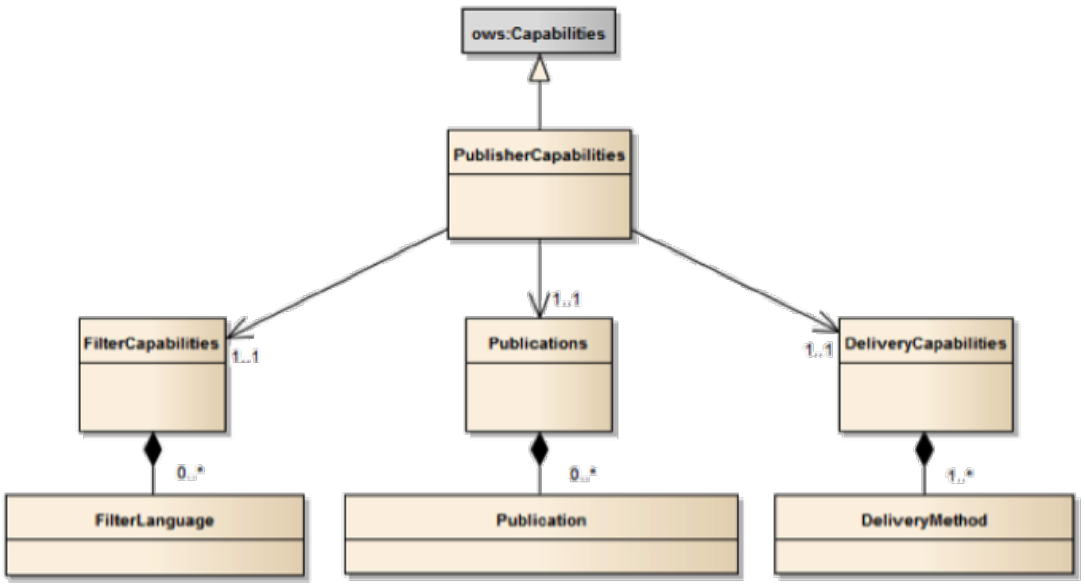


Figure 8. PubSub Capabilities components

7.2.1. FilterCapabilities

The FilterCapabilities section describes the filtering-related capabilities of a PubSub-OWS, i.e. the filter languages it supports for matching events against subscriptions (e.g., OGC Filter Encoding). This allows the pluggability of filter languages.

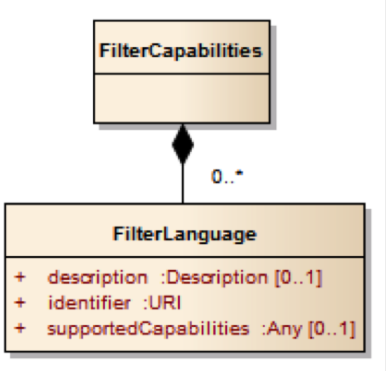


Figure 9. Filter Capabilities

The SupportedCapabilities elements allows restricting the acceptable requests, possibly providing templates. The following Capabilities snippet declares that this PubSub-OWS instance (namely, a CSW) accepts as subscription filters GetRecords requests conforming to the specified OpenSearch template. Multiple templates may be introduced, specifying multiple FilterLanguages.

FilterCapabilities

```
<FilterCapabilities>
   <FilterLanguage>
      <Abstract>This PubSub-OWS accepts requests as subscription filters, according to
the OpenSearch template specified in SupportedCapabilities.
      </Abstract>
      <Identifier>http://www.opengis.net/spec/pubsub/1.0/conf/ows/request-reply-
publisher</Identifier>
      <SupportedCapabilities>http://tb12.essi-lab.eu/pubsub-
csw/services/opensearch?ct={count?}&amp;st={searchTerms?}&amp;bbox={geo:box?}&amp;ts={
time:start?}&amp;te={time:end?}
      </SupportedCapabilities>
   </FilterLanguage>
</FilterCapabilities>
```

7.2.2. DeliveryCapabilities

The DeliveryCapabilities section describes the delivery methods supported by the PubSub-OWS, e.g. SOAP, WS-Notification, ATOM, SSE, WebSockets, OAI-PMH. This allows the pluggability of delivery methods.

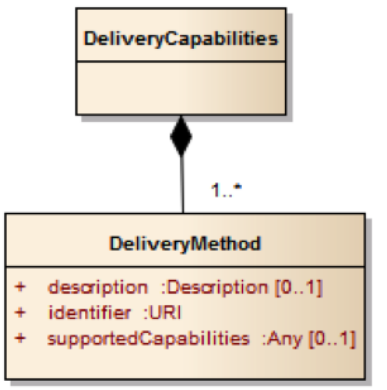


Figure 10. Delivery Capabilities

The following Capabilities snippet declares that this PubSub-OWS instance delivers notifications via SSE (see chapter Delivery methods, below).

DeliveryCapabilities

```
<DeliveryCapabilities>
   <DeliveryMethod>
      <Abstract>This PubSub-OWS supports notification delivery via SSE.
      </Abstract>
      <Identifier>http://www.w3.org/TR/eventsource/
      </Identifier>
   </DeliveryMethod>
</DeliveryCapabilities>
```

Delivery methods

The DeliveryCapabilities section describes the methods supported by the PubSub-OWS for delivering notifications. Publishers may offer more than one method of delivery for each Publication, to be chosen by Subscribers. Publish/Subscribe would imply push-style message delivery, however some methods may actually be pull-based (e.g. polling), under the hood.

Examples include: SOAP and related technologies, such as WS-Notification (used by PSSB), ATOM (polling using the “If-Modified-Since” and “start-index” parameters), PubSubHubbub, OAI-PMH (polling using the “from” parameter), e-mail, SMS, WebSockets, SSE.

Server-Sent Events (SSE) is a pure push-style communication technology based on HTTP and the SSE EventSource API standardized as part of HTML5 by the W3C. A SSE client (e.g. all modern HTML 5.0 browsers) receives automatic updates from a server via HTTP connection, simply setting the following parameters:

- ContentType: "text/event-stream;charset=UTF-8”
- Cache-Control: "no-cache”
- Connection: "keep-alive”

7.2.3. Publications

The Publications section describes the contents offered by the PubSub-OWS, i.e. the sequences of notifications that Subscribers can subscribe to.

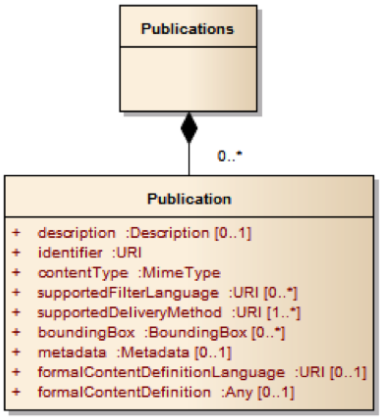


Figure 11. Publications

The following Capabilities snippet declares a publication that notifies on all the relevant events for this PubSub-OWS. Notifications can be filtered with the semantics of the requests of this OWS and are delivered via SSE, encoded in JSON (see chapter Notification encoding, below).

Publications

```
<Publications>
   <Publication>
      <Abstract>>This publication notifies on all the relevant events for this PubSub-
OWS.
      </Abstract>
      <Identifier>ALL</Identifier>
      <ContentType>application/json</ContentType>

<SupportedFilterLanguage>http://www.opengis.net/spec/pubsub/1.0/conf/ows/request-
reply-publisher</SupportedFilterLanguage>

<SupportedDeliveryMethod>http://www.w3.org/TR/eventsource/</SupportedDeliveryMethod>
   </Publication>
</Publications>
```

Notification encoding

For the generic OWS instance, no operation is defined that provides the basic semantics of “insert”, “update”, and “delete” actions on the content managed by the instance.

The most generic mechanism to notify about updates is that the Publisher re-send the whole response element corresponding to the request used as filter in the Subscription. For example, in the case of WFS, if the client subscribes with a wfs:GetFeature request as a filter, the PubSub-WFS should notify about any changes by delivering a standard wfs:FeatureCollection, in response to that request.

By receiving the new response and comparing it with the previous one, a Subscriber can figure out the changes. Future evolutions of this extension may evaluate more efficient and semantically accurate encoding of notifications. A possible option for XML-based content types is XMLdiff (e.g. XML Patch, RFC 5261), or annotations (XML attributes) to add simple CRUD semantics on top of the

existing XSDs.

7.3. Support to legacy components

The integration of legacy components in an eventing architecture is desirable in a number of scenarios. However, legacy components may not be instrumented to monitor their state for the purpose of notification, nor to react upon notifications from other components (or they may, but by legacy, non-standard mechanisms).

Implementing the PubSub 1.0 Standard in a legacy component may not be feasible or practical. In some cases, the legacy component can be adapted to the Publish/Subscribe MEP by an additional functional entity that realize the Publish/Subscribe functionalities. Such mediating entity acts as a proxy/adapter, i.e. a middleman between the source and the target of the message exchange, implementing the behavior and/or the interfaces required by the PubSub specification.

This use case has been considered in the phase of requirement analysis for the PubSub 1.0 standard [3: See also the Proxied Publish/Subscribe use case (access restricted to OGC Members): https://portal.opengeospatial.org/wiki/PUBSUBswg/PubSubSwgUseCaseBrokeredPubSub] and is supported by the Brokering Publisher Conformance Class of the PubSub 1.0 Standard.

Depending on the intended role of the legacy component, the use case is twofold:

- Proxied Subscribe – a proxy/adapter component subscribes to a Publisher on behalf of the legacy system and acts appropriately upon receiving notifications of interest.

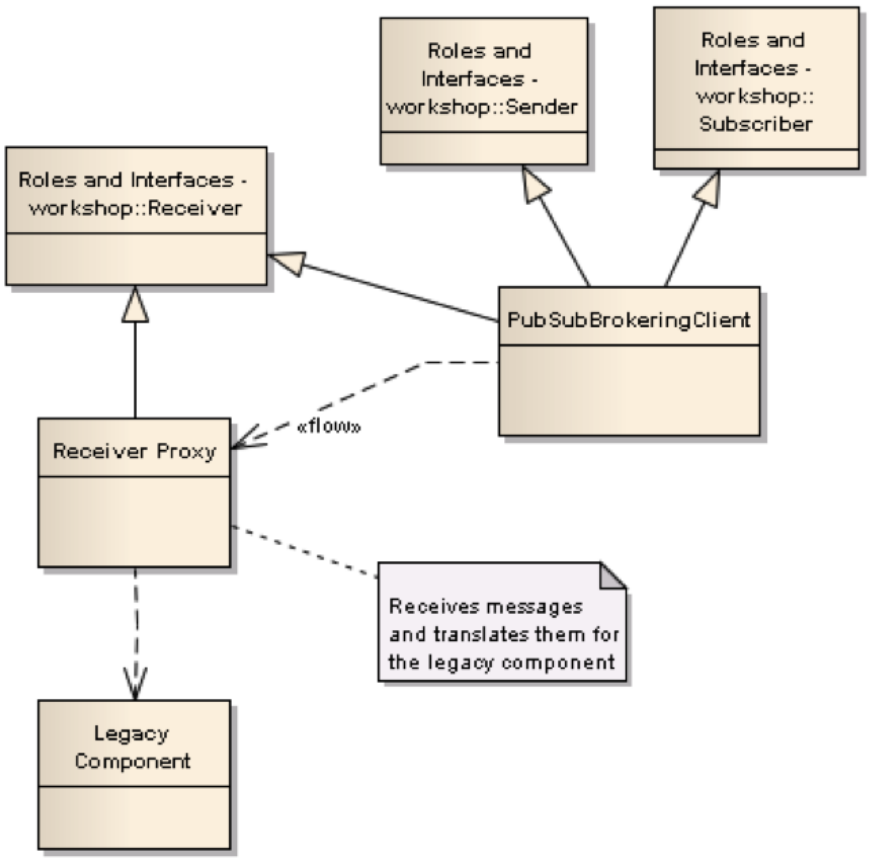


Figure 12. Proxied subscribe

- Proxied Publish – a proxy/adapter component monitors the legacy system and generates

appropriate notifications upon relevant events (according to a given event model). The proxy/adapter may act as a full-fledged Publisher, accepting Subscriptions against the sequence of notifications, or just act as a pure Sender, relaying each notification to another Publisher entity (see figure Figure 10).

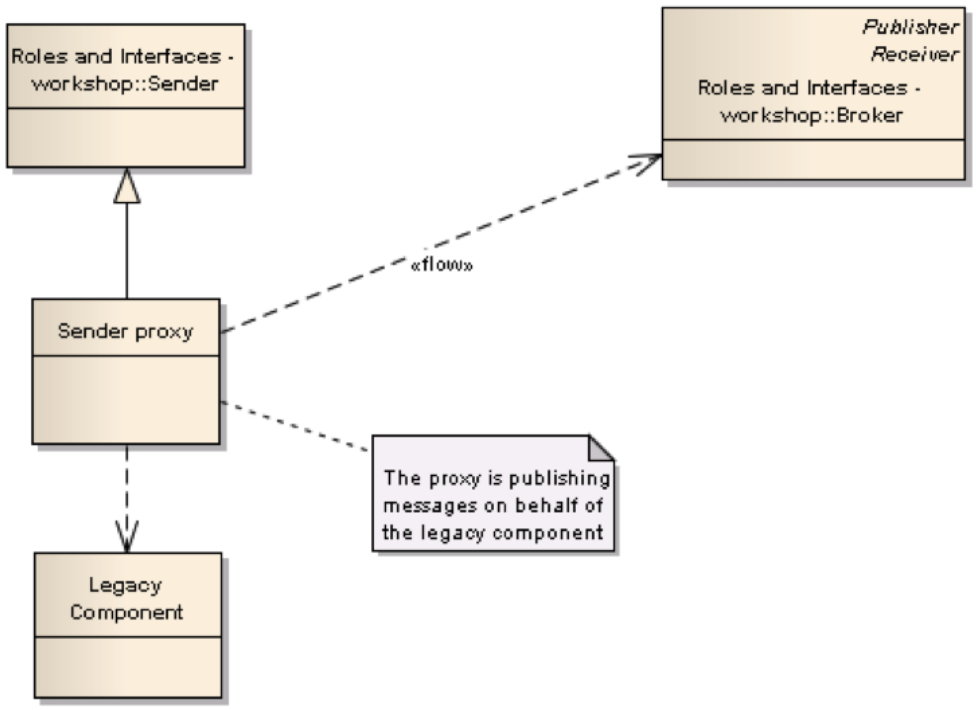


Figure 13. Proxied publish

The Brokering Publisher Conformance Class of the PubSub 1.0 Standard supports this use case. In fact, a Brokering Publisher (or, more simply, a broker), is an intermediary between Subscribers and other Publishers which have been previously registered with the broker. The broker is not the original producer of messages, but only acts as a message middleman, re-publishing messages received from other Publishers and decoupling them from their Subscribers. A broker may shuffle or aggregate messages into different publications, may offer publications with different delivery methods than the original ones, or otherwise process the messages (e.g. converting their format). A broker may also provide advanced messaging features, such as load balancing.

In general, a broker is a distinct third party that acts as a communication intermediary between the source and the target of a communication, mediating their interfaces and in some cases adding new behavior. Hence, a broker may conveniently act as a proxy/adapter for one or more legacy components, flexibly implementing any combination of the above twofold use case.

The Brokering Publisher Conformance Class does not mandate any specific behavior to be implemented, in particular as regards the support to Delivery Capabilities, Filtering Capabilities, and Publications of the brokered Publishers. Brokers are free to interact with the brokered Publishers as appropriate for their specific application. Interactions may include subscribing to the offered publications, harvesting the data, decorating the capabilities, or other behavior (future extensions of the Conformance Class may standardize the behavior of Brokering Publishers in specific application scenarios).

Examples of Brokering Publisher applications include the following:

- Publisher Aggregation – a broker subscribes to several Publishers and relays their publications

(without modifications) to interested Subscribers, acting like a Proxy to multiple Publishers. Optionally, the broker may adapt the service interface (binding) of the aggregated Publishers.

- Publication Aggregation – a broker receives messages generated by several Publishers (e.g. dumb sensors) and publishes them to the interested Subscribers as a single publication at a single endpoint, for the sake of simpler connectivity, or improved accountability, or easier management of subscriptions, etc.
- GeoSynchronization (GSS) - GSS is a mediation service that controls transactional access to one or more WFS's (e.g. to moderate updates in crowdsourcing scenarios). A GSS maintains several event channels, including one for changes applied to the WFS content. Clients can subscribe to the channels (possibly specifying a filter) and be notified by the GSS whenever new entries appear. A GSS may be used to monitor insert/update/delete operations performed on one or more WFS's and send appropriate notifications, implementing the PubSub 1.0 Brokering Publisher Conformance Class. Whenever an event (i.e. a Transaction) occurs on a WFS, the GSS will notify Subscribers of that event. In this way WFS's that do not implement the PubSub 1.0 Standard can participate in an eventing architecture. There are plans to extend GSS to other OGC access services, such as WCS.

Chapter 8. Report on the PubSub RESTful Binding

The PSRB specifies how PubSub 1.0 Core operations, encodings and messages are bound to a RESTful service interface. The specification is currently in draft state and is shared on the PubSub SWG online folders, as document OGC 13-132. We have assessed, improved and experimented the current PSRB specification, implementing a PubSub-enabled CSW with a RESTful interface.

To maximize our contribution to the work of the PubSub Standards Working Group (SWG), we agreed with the Testbed-12 management to align with the SWG toolchain, annotating our comments and observations in the PSRB document draft itself. This way, the draft has always remained available to the PubSub SWG, which has also been identified as the primary Working Group to review this ER.

The main findings from this activity are briefly summarized in the following:

- Consolidated design of the PubSub RESTful API interface (note: [endpoint] is the service instance RESTful endpoint; the trailing '/' is optional):

Endpoint	**Resource**	**Method**	**Behavior**	**Request entity**
[endpoint]/	PubSub capabilities	GET	Return capabilities document	-
[endpoint]/subscriptions	Collection of (active?) subscriptions created by user	GET	Return subscription list	-
		POST	Create subscription	application/x-www-form-urlencoded, application/xml
[endpoint]/subscriptions/{id}	Subscription	GET	Return subscription	-
		PUT	Update (replace) subscription	application/xml
		PATCH	Partial update of subscription	application/xquery (XQUF)
		DELETE	Delete subscription (unsubscribe)	-

- PATCH method not recommended because of unclear semantics and scarce support in implementations (e.g. Java JAX-RS annotations);
- OPTION method recommended for resource browsing/discovery;
- HEAD method recommended for implementing security mechanisms (see also A065);
- Implemented RESTful Basic Publisher on top of a CSW-ISO instance, offering publications in JSON and delivering notifications via Server-Sent Events (SSE).

Appendix A: XML Schema Documents

No additional XML Schema Documents are introduced in this Engineering Report.

Appendix B: Relevant findings

- OGC 13-131, OGC® Publish/Subscribe Interface Standard 1.0 - Core
  - Typos in Basic Publisher requirement tables (§8, p. 22). These typos were fixed directly in the document, before its first publication.

B.1. Recommendations

- The following activities may be considered in defining Testbed 13 requirements, mainly as part of Work Items 2 and 3: [4: http://www.opengeospatial.org/projects/initiatives/testbed13/#_work_items]
  - **PubSub-broker**: the Brokering Publisher Conformance Class may be used to integrate legacy components (not PubSub-enabled) into an event-based scenario. The activity may implement a proxy component, implementing the behavior and/or the interface required by the PubSub specification, and acting as a middleman between sources and targets of message exchanges (see also chapter Legacy Components);
  - **PubSub-enabled OWS's** (other than Catalogs): e.g. access services like WCS, to notify subscribers when new coverages are made available; or portrayal services like WMS. The activity may regard the implementation of the PubSub extension for the generic OWS introduced in chapter OWS PubSub Extension, or the definition of specific PubSub conformance classes, like the one introduced for CSW in chapter CSW PubSub Extension);
  - **Machine-to-machine Publish/Subscribe** (M2M-PubSub): when subscribers are services themselves, instead of human users, advanced use-cases may be implemented involving an interconnected and interchangeable set of loosely coupled actors (e.g. an analyst gets imagery on a given region of interest, anytime it becomes available, by means of a combination of Catalog, Web Event Processing, and Web Coverage Services). The activity may also explore the interoperability of OGC PubSub with other orchestration technologies (cf. choreography, workflows, WebHooks, etc.);
- For the PubSub SWG:
  - Evaluate the Basic PubSub 1.0 extension for the generic OWS as a basis for a Best Practice Paper;
  - Consider incorporating (part of) chapter PSRB Report in the current Publish/Subscribe RESTful Binding (PSRB) draft.
- For the Catalog DWG and the Catalog Services 3.0 SWG:
  - Evaluate the Specific PubSub 1.0 extension for the CSW as a basis for a Best Practice Paper.

B.2. Use Case(s)

The main use-case addressed by the extensions introduced in this document is the notification when new resources match the client's criteria of interest, as specified in the client's subscription. Additional use-cases may include: the deletion or the modification of existing resources; the

modification of the Capabilities document itself; or changes in other pieces of information retrievable from an OWS. Such use-cases may be investigated in a future edition of the Testbed Initiative, or by the individual Working Groups interested.

B.3. Architectural schemes

None relevant.

B.4. Change Requests

- OGC 13-131, OGC® Publish/Subscribe Interface Standard 1.0 - Core
  - Figure 1 shows a dependency of Basic Publisher from Basic Receiver, which is probably incorrect and should be deleted;
  - The title of chapter 15 reads "Requirements Class - Capabilities Filtering extends Basic Publisher", whereas it should read "Requirements Class - Capabilities Filtering extends Standalone Publisher";
  - Add paragraph on GetCapabilities to the PublicationManager class (e.g. §14.2) and make §14.1.1 (ProcessingCapabilities) a sub-paragraph, in analogy to §13.3 and §13.3.1;
  - Figure 1 may detail the dependency of Basic Publisher on OWS clauses 8, 10, and Standalone Publisher on OWS clause 7.
- OGC 06-121r3, OGC® Web Services Common Specification, version 1.1.0 (9 February 2007)
  - As of OWS Common 1.1, an OWS can advertise the implemented conformance classes in its Capabilities document, namely in the Profile property of the ServiceIdentification section. However, some specifications (e.g. CSW 2.0.2) depend on OWS Common 1.0.0, which does not define the Profile property in ServiceIdentification. An alternative mechanism should be recommended to work around this issue, e.g. including an additional OWS Common 1.1 ServiceIdentification block in the ExtendedCapabilities;
  - The PubSub Core 1.0 Standard requires additional Capabilities components to be returned in the GetCapabilities response, but does not specify the specific mechanism for incorporating these additional Capabilities components into the OWS Capabilities document. A specific mechanism may be defined, e.g. including these additional Capabilities components in the ExtendedCapabilities.

Appendix C: Prototype PubSub-CSW implementation

Some of the solutions proposed in the ER have been experimented by CNR in the context of the Testbed 12 A016 component deliverable: CSW 2.0.2 with PubSub Core Support.

The prototype instance [5: http://tb12.essi-lab.eu/pubsub-csw] implements CSW-ISO (OGC 07-006r1, 07-045) and features a PubSub adapter that provides a RESTful API satisfying the requirements for a Basic Publisher. It offers publications in JSON, and delivers notifications via Server-Sent Events (SSE). An associated prototype client (in-kind contribution) may be used to demonstrate it. The following chapters describe specific design aspects of the implementation and provide examples of publish/subscribe messages.

C.1. Publications

This instance offers the following publications in the JSON format, over a SSE stream:

- EARTHQUAKE - notifications on new records from the USGS Earthquake Catalog, an implementation of the FDSN Event Web Service Specification, allowing custom searches for earthquake information using a variety of parameters. As described in the capabilities document, this publication supports only the spatial filter;
- RANDOM - random notifications forged for testing convenience, generated on regular frequency (configured from GI-cat configuration manager; default is every 1 minute). Each notification contains 1-5 records; the content of "reports" is a constant string (see the example below). As described in the capabilities document, this publication supports only the empty filter;
- ROOT - notifications on new records added to the whole record set of this catalog. As described in the capabilities document, this publication supports only the empty filter.

Note: a previous version of the service offered the following publication (now deprecated by IRIS): IRIS - notifications on new records from the IRIS DMC FDSNWS Event Web service. The IRIS Event service aggregates event data from a number of independently-operated catalogs of seismic events.

The following is an example of a notification, as shown by Chrome opening the DeliveryLocation URL:

```
event: e91930
retry: 300000
data: {"result":"{\"reports\":[{\"description\":\"The
abstract\",\"topic\":[\"documentFileGraphic\"],\"where\":[{\"east\":96,\"south\":-
30,\"north\":36,\"west\":-168}],\"id\":\"b6d247cd-aae0-4f14-a14b-
ba369fde0fef\",\"type\":\"simple\",\"title\":\"Random dataset b6d247cd-aae0-4f14-a14b-
ba369fde0fef\",\"keyword\":[\"Another keyword\",\"A
keyword\"],\"harvested\":true,\"geossCategory\":[\"documentFileGraphic\"]},{\"descript
ion\":\"The
abstract\",\"topic\":[\"documentFileGraphic\"],\"where\":[{\"east\":116,\"south\":-
90,\"north\":0,\"west\":-61}],\"id\":\"e8ec62ba-8de4-4fd9-bad5-
b248eefa8d7d\",\"type\":\"simple\",\"title\":\"Random dataset e8ec62ba-8de4-4fd9-bad5-
b248eefa8d7d\",\"keyword\":[\"Another keyword\",\"A
keyword\"],\"harvested\":true,\"geossCategory\":[\"documentFileGraphic\"]}],\"resultSe
t\":{\"pageCount\":1,\"size\":2,\"pageIndex\":1,\"start\":1,\"pageSize\":10}}","query"
:{"json":"http://tb12.essi-lab.eu/pubsub-csw/services/opensearch?reqID=43d77f&ts=2015-
01-01&te=2015-09-01&bbox=-10,-
10,10,10&st=water&parents=&sources=&outputFormat=application/json&from=1465996958756&u
ntil=1465997022431","atom":"http://tb12.essi-lab.eu/pubsub-
csw/services/opensearch?reqID=43d77f&ts=2015-01-01&te=2015-09-01&bbox=-10,-
10,10,10&st=water&parents=&sources=&outputFormat=application/atom+xml&from=14659969587
56&until=1465997022431"},"from":1465996958756,"until":1465997022431,"updates":"2"}
```

Note that:

- the "event" field corresponds to the subscription identifier (in this case "e91930");
- the "retry" field instructs the browser to attempt a new connection (in case of disconnection) after 5 minutes (300.000 milliseconds);
- the "data" field contains a JSON object with the following properties:
  - "result": this object contains the "reports" array, which in turn contains:
    - the records which satisfy the subscription filter, 2 in this example (see this link for more info about the "reports" field);
    - a "resultSet" object (see this link for info about the "resultSet");
  - "query": this object contains the URL to retrieve the updates from the GI-cat OpenSearch interface in JSON or ATOM;
  - "from" and "until": the time extent of the updates expressed in Unix time;
  - "updates": the number of updates (corresponds to the length of the "reports" array).

C.2. Subscribe

C.2.1. Subscribe request

The following is an example of Subscribe request on the ROOT publication with an empty filter.

```
POST /pubsub-rest/subscriptions
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<pubsub:Subscribe xmlns="http://www.opengis.net/pubsub/1.0"
   xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gco="http://www.isotc211.org/2005/gco"
   xmlns:gml="http://www.opengis.net/gml/3.2"
xmlns:ows="http://www.opengis.net/ows/1.1"
   xmlns:gts="http://www.isotc211.org/2005/gts"
xmlns:gmd="http://www.isotc211.org/2005/gmd"
   xmlns:gsr="http://www.isotc211.org/2005/gsr"
xmlns:gss="http://www.isotc211.org/2005/gss"
   xmlns:pubsub="http://www.opengis.net/pubsub/1.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <pubsub:PublicationIdentifier>ROOT</pubsub:PublicationIdentifier>
   <pubsub:FilterLanguageId>empty_filter</pubsub:FilterLanguageId>
   <pubsub:Filter>http://tb12.essi-lab.eu/pubsub-
csw/services/opensearch?outputFormat=application/json</pubsub:Filter>
</pubsub:Subscribe>
```

Note that:

- the filter is empty in this case, but may contain parameters (see the table below, and the "complete filter" in the Capabilities document); if so, the parameters must be separated by the URL-encoded "&" character (otherwise the server throws a parsing exception);
- the "Content-Type" HTTP header must be set in the request, with the value "application/xml";
- this instance publisher allows a maximum of 50 subscriptions; this value can be configured in the GI-cat configuration manager;
- this instance publisher is configured so that subscriptions expire in one hour, by default. A different expiration time may be requested by adding the following element to the request (in this case, the expiration would be set at the end of September 2016):

```
<pubsub:Expiration gml:id="6046bd">
     <gml:timePosition>2016-09-30T00:00Z</gml:timePosition>
</pubsub:Expiration>
```

The following table describes all the filter parameters theoretically supported by this instance publisher (the actual subset depend on the publication, as indicated in the Capabilities).

Parameter name	**Parameter description**
st	One or more search terms. In case of multiple search terms they can be separated by the "AND" conjunction. E.g: "water", "water AND snow"

Parameter name	Parameter description
bbox	A bounding box expressed in lon/lat EPSG:4326: west, south, east, north. For more info see the OpenSearch Geo extension documentation of the "box" parameter
ts	The beginning of the time slice of the search query. For more info and examples see OpenSearch Time extension documentation of the "start" and "end" parameters
te	The end of the time slice of the search query. For more info and examples see OpenSearch Time extension documentation of the "start" and "end" parameters

C.2.2. Subscribe response

The relevant part of the Subscribe response is the DeliveryLocation. A browser supporting Server-Sent Events (SSE) technology, such as Chrome, can directly open the URL in the DeliveryLocation, and listen to the generated events. In Chrome the events are showed directly in the web page (see next section for an example), while other browsers like Firefox just open the stream. However, the best way to manage Server-Sent events is by creating an EventSource Javascript object passing as argument the DeliveryLocation URL.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SubscribeResponse xmlns="http://www.opengis.net/pubsub/1.0"
xmlns:ns2="http://www.opengis.net/gml/3.2" xmlns:ns3="http://www.w3.org/1999/xlink"
xmlns:ns4="http://www.opengis.net/ows/1.1">
  <Subscription>
    <SubscriptionIdentifier>e91930</SubscriptionIdentifier>
    <PublicationIdentifier>ROOT</PublicationIdentifier>
    <Expiration ns2:id="6046bd">
         <ns2:timePosition>2030-09-29T15:19Z</ns2:timePosition>
    </Expiration>
    <FilterLanguageId>empty_filter</FilterLanguageId>
    <Filter xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string">http://tb12.essi-
lab.eu/pubsub-csw/services/opensearch?outputFormat=application/json</Filter>
    <DeliveryLocation>http://tb12.essi-lab.eu/pubsub-
csw/services/pubsub?outputFormat=application%2Fjson&amp;client=js&amp;request=subscrib
e&amp;label=OGC_PubSub_subscription_df45ca&amp;clientID=CSWPublisher&amp;subscriptionI
D=df45ca&amp;init=true&amp;creation=1465991415263&amp;expiration=1916925540000&amp;req
ID=096253</DeliveryLocation>
  </Subscription>
</SubscribeResponse>
```

Note that:

- since the Subscribe response is encoded in XML, the DeliveryLocation URL is URL-encoded. In order to avoid possible parsing problems by browsers or other tools/applications, the recommended practice is to replace the URL-encoded "&" with the "&" character; for example, in Chrome the URL with URL-encoded "&" is accepted without errors, but the SSE event stream seems not to be opened
- each delivery location can only be consumed once. So, for example, opening the same delivery location in two different Chrome tabs will result in a "SUBSCRIPTION_REJECTED" error:

```
event: error
data: {"type":"SUBSCRIPTION_REJECTED"}
```

C.3. GetSubscription

C.3.1. GetSubscription request

The GetSubscription request is performed with a GET method on the following paths:

- path to get subscription with the given identifier:

```
/pubsub-rest/subscriptions/{subscriptionIdentifier}
```

- path to get all the active subscriptions:

```
/pubsub-rest/subscriptions/
```

Example of GetSubscription request:

```
http://tb12.essi-lab.eu/pubsub-csw/services/pubsub-rest/subscriptions/e91930
```

C.4. GetSubscription response

Example of GetSubscription response:

```
<GetSubscriptionResponse xmlns="http://www.opengis.net/pubsub/1.0"
 xmlns:ns2="http://www.opengis.net/gml/3.2" xmlns:ns3="http://www.w3.org/1999/xlink"
 xmlns:ns4="http://www.opengis.net/ows/1.1">
 <Subscription>
 <SubscriptionIdentifier>e91930</SubscriptionIdentifier>
 <PublicationIdentifier>ROOT</PublicationIdentifier>
 <Expiration ns2:id="6046bd">
 <ns2:timePosition>2030-09-29T15:19Z</ns2:timePosition>
 </Expiration>
 <FilterLanguageId>empty_filter</FilterLanguageId>
 <Filter xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xs="http://www.w3.org/2001/XMLSchema" xsi:type="xs:string"
 >http://tb12.essi-lab.eu/pubsub-
csw/services/opensearch?outputFormat=application/json</Filter>
 <DeliveryLocation>http://tb12.essi-lab.eu/pubsub-
csw/services/pubsub?outputFormat=application%2Fjson&client=js&request=subscribe&label=
OGC_PubSub_subscription_df45ca&clientID=CSWPublisher&subscriptionID=df45ca&init=true&c
reation=1465991415263&expiration=1916925540000&reqID=096253</DeliveryLocation>
 </Subscription>
</GetSubscriptionResponse>
```

C.5. Unsubscribe

C.5.1. Unsubscribe request

The Unsubscribe request is performed with a DELETE method on the following path:

```
/pubsub-rest/subscriptions/{subscriptionIdentifier}
```

Example of Unsubscribe request:

```
http://tb12.essi-lab.eu/pubsub-csw/services/pubsub-rest/subscriptions/e91930
```

C.5.2. Unsubscribe response

If the request is accepted and no errors occur, the Unsubscribe response is returned and the notifications terminate with the "close" event. Example of Unsubscribe response:

```
<pubsub:UnsubscribeResponse xmlns:pubsub="http://www.opengis.net/pubsub/1.0">
</pubsub:UnsubscribeResponse>
```

Example of SSE close event:

```
event: close
```

Appendix D: Revision History

Table 2. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
April 15, 2016	L. Bigagli	r0	all	Initial ER
June 30, 2016	L. Bigagli	r1	all	First full-draft ER
October 7, 2016	L. Bigagli	r2	all	Draft ER
October 7, 2016	M. Lawrence	r2	CSW 3.0	
October 31, 2016	L. Bigagli	r3	all	Addressed IP Team comments, finalisation
November 15, 2016	M. Lawrence	r4	CSW 3.0	
November 15, 2016	P. Vretanos	r4	Service Interface	
November 15, 2016	L. Bigagli	r4	all	Final Draft ER Addressed NGA comments, final revision
November 21, 2016	L. Bigagli	r5	Relevant findings, cross-links, examples	Typos and fixes
December 19, 2016	L. Bigagli	r5	Overview	Addressed further comment by IP Team and NGA on relationship with ER A067

Appendix E: Bibliography

No relevant bibliography entry.